PATENT
5760-16400
VRTS 0502

"EXPRESS MAIL" MAILING LABEL NUMBER
EV 318248480 US
DATE OF DEPOSIT 11-12-03
I HEREBY CERTIFY THAT THIS PAPER OR FEE IS BEING DEPOSITED WITH THE UNITED STATES POSTAL SERVICE "EXPRESS MAIL POST OFFICE TO ADDRESSEE" SERVICE UNDER 37 C.F.R. §1.10 ON THE DATE INDICATED ABOVE AND IS ADDRESSED TO THE COMMISSIONER FOR PATENTS, MAILSTOP PATENT APPLICATION, P.O. BOX 1450, ALEXANDRIA, VA 22313-1450

Derrick Brown

Provisioning and Snapshotting Using Copy on Read/Write and Transient Virtual Machine Technology

By:

Hans F. van Rietschote
Mahesh P. Saptarshi
Craig W. Hobbs

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention is related to the field of provisioning computer systems to execute a set of software resources.

2. Description of the Related Art

Software provisioning of computer systems is becoming more pervasive as businesses attempt to more fully utilize the computer power they have at their disposal. As the workload being performed by a set of computer systems changes, it may be desirable to shift workload of various types from one computer system to another, to increase or decrease the number of computer systems handling a given type of workload, etc. Different workloads may require different software resources (e.g. operating systems, applications, etc.), and thus the configuration of the computer systems may need to be changed. Providing a computer system with a set of software resources is referred to as provisioning the computer system.

Software provisioning generally involves copying an image of the software resources from an image repository system to the target computer system, typically across a network of some sort. Provisioning a computer system frequently takes a relatively long period of time (e.g. 20 minutes or more, depending on the size of the image, the speed of the network, etc.). The amount of time required to provision a computer system may impact how efficiently computer systems can be redeployed to handle other workloads. For example, the OpForce™ product currently available from VERITAS Software Corporation (Mountain View, CA) may do the following to provision a computer system (OpForce executes on the image repository system): The target computer system sends out a Pre Execution Environment (PXE) boot request (during boot of the target computer system). OpForce receives the PXE boot request, and transmits a lightweight Linux operating system (the "Active OS") to the target computer system. The

target computer system boots into the Active OS, which mounts the image repository system containing the image to be provisioned (e.g. a network file system (NFS) or common internet file system (CIFS) mount). The Active OS executes a make filesystem command on the target computer system, and copies all files from the image on the image repository system to the local disk in the target computer system The Active OS "personalizes" the local files (hostname, IP address, etc). The Active OS then reboots the target computer system. The target computer system again sends out a PXE boot request, which OpForce ignores. The target computer system times out on the PXE boot request and boots from the local disk.

In some cases, provisioning may be accomplished by having multiple bootable images in a storage area network (SAN) or Network Attached Storage (NAS), and directing a target computer system to boot from the desired image. Small Computer Systems Interface over Internet Protocol (iSCSI) disks could be used in a similar fashion. While some high-end server systems may be capable of such booting, many computer systems (e.g. personal computer (PC) systems based on Intel or AMD processors) typically are not designed to boot in this fashion. If a large amount of local storage is included in a target computer system, multiple bootable images may be stored in the local storage, and the desired image may be selected for booting (e.g. so called "multiple bootable partitions").

Another function which may take a relatively long period of time is snapshotting a system image from a computer system into the image repository. For example, the current OpForce product may perform the following to snapshot an image into the image repository system: OpForce communicates with an OpForce agent running on the target computer system, requesting a reboot of the target computer system. The target sends out a PXE boot request during the reboot. OpForce detects the PXE boot request, and pushes the Active OS to the target computer system in response. The target computer system boots the Active OS, which mounts the image repository system. Active OS copies the

changed files to the image repository system. Determining which files have changed requires communication back and forth between the target computer system and the image repository system (e.g. comparing time stamps of corresponding blocks on the target computer system and the image repository system).

During both provisioning and snapshotting, the target computer system may be unavailable to perform other work.

SUMMARY OF THE INVENTION

In one embodiment, a computer accessible medium comprises instructions which, when executed, check a first storage from which a computer system is configured to boot for a block identified in a read request generated on the computer system. The block is included within an image of a set of software resources to be provisioned on the computer system. If the block is stored in the first storage, the instructions supply the block from the first storage in response to the read request. If the block is not stored in the first storage, the instructions: fetch the block from an image repository system that stores the image; store the block in the first storage; and supply the block in response to the read request.

In another embodiment, a computer accessible medium comprises instructions which, when executed, store a block identified in a write request generated within a computer system to a first storage from which the computer system is configured to boot. The block is included in an image of a set of software resources that are provisioned on the computer system. The instructions also record that the block is modified with respect to the image stored in an image repository system.

In yet another embodiment, a method for provisioning a computer system with a set of software resources is contemplated. Execution of the set of software resources is initiated on the computer system prior to storing at least some blocks comprising the set

of software resources to a first storage from which the computer system is configured to boot. The computer system generates a read request for a first block of the blocks comprising the set of software resources. If the first block is stored in the first storage, the first block is supplied from the first storage. If the first block is not stored in the first storage: the first block is fetched from an image repository system that stores an image of the set of software resources; the first block is stored in the first storage; and the first block is supplied in response to the read request.

In another embodiment, an apparatus comprises an image repository system and a computer system. The image repository system is configured to store an image of a set of software resources. The computer system is configured to transmit a remote boot request in response to booting. The image repository system is coupled to receive the remote boot request, and is configured to detect that the computer system is to be provisioned with the set of software resources. The image repository system is configured to respond to the remote boot request with a program which, when executed by the computer system, initiates execution of the set of software resources prior to at least some blocks in the image being copied to the computer system.

In still other embodiments, an apparatus comprises an image repository and a computer system. The image repository is configured to store an image of a set of software resources. The computer system is configured to execute the set of software resources, and is configured to track blocks: (i) in a first storage from which the computer system is configured to boot; and (ii) corresponding to the image which are updated with respect to the image stored on the image repository. The updates are generated by the computer system during execution of the set of software resources. A snapshot of the image as updated by the computer system is generated by transmitting the modified blocks from the computer system to the image repository system and not transmitting unmodified blocks.

BRIEF DESCRIPTION OF THE DRAWINGS

The following detailed description makes reference to the accompanying drawings, which are now briefly described.

Fig. 1 is a block diagram of one embodiment of a target system and an image repository system.

Fig. 2 is a flowchart illustrating one embodiment of a method of provisioning the target system.

Fig. 3 is a block diagram of one embodiment of the target system and the image repository at a first point in time during the provisioning.

Fig. 4 is a block diagram of one embodiment of the target system and the image repository at a second point in time during the provisioning.

Fig. 5 is a flowchart illustrating operation of one embodiment of a device driver for the local storage of the target system in response to a block read request.

Fig. 6 is a flowchart illustrating operation of one embodiment of a background provisioner shown in Fig. 4.

Fig. 7 is a flowchart illustrating one embodiment of a method for snapshotting a provisioned target system.

Fig. 8 is a flowchart illustrating operation of one embodiment of the device driver for the local storage of the target system in response to a block write request.

Fig. 9 is a flowchart illustrating operation of one embodiment of a background

replicator shown in Fig. 4.

Fig. 10 is a block diagram of one embodiment of a local storage bitmap shown in Figs. 3 and 4.

Fig. 11 is a block diagram of one embodiment of a computer accessible medium.

While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof are shown by way of example in the drawings and will herein be described in detail. It should be understood, however, that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the intention is to cover all modifications, equivalents and alternatives falling within the spirit and scope of the present invention as defined by the appended claims.

DETAILED DESCRIPTION OF EMBODIMENTS

Turning now to Fig. 1, a block diagram is shown illustrating one embodiment of a target computer system 10 and an image repository computer system 12 coupled to a network 14. The target computer system 10 may be more succinctly referred to herein as the target system 10 or system 10, and the image repository computer system 12 may be more succinctly referred to as the image repository system 12 or system 12. While one target system 10 and one image repository system 12 are shown in Fig. 1, generally one or more target systems 10 and one or more image repository systems 12 may be included in various embodiments. Each of the systems 10 and 12 may include execution hardware (e.g. the execution hardware 16A in the system 10 and the execution hardware 16B in the system 12) and local storage coupled to the execution hardware (e.g. the local storage 18A in system 10 and the local storage 18B in system 12). The image repository system 12 may further include provisioner software 20. Additionally, the local storage 18B may store one or more images including images 22A-22B shown in Fig. 1. Each image 22A-

22B may include one or more image blocks. For example, the image 22A is illustrated as containing image block 0 (IB0 in Fig. 1), image block 1 (IB1 in Fig. 1), and other image blocks up to image block N (IBN in Fig. 1).

The target system 10 may be a computer system that is to be provisioned with a set of software resources included within one of the images 22A-22B on the image repository system 12. For this discussion, the image 22A may include the software resources to be provisioned on the target system 10, although generally any image may be used. As used herein, the term "software resources" includes at least an operating system (referred to herein as the "target operating system"). Additionally, the software resources may include one or more applications to be executed on the target operating system. The software resources may, in some embodiments, further include data structures used by the target operating system and/or the applications. The software resources may, in some embodiments, also include various configuration parameters for the hardware (e.g. the IP address, hostname, etc. mentioned above as well as enabling or disabling various hardware or hardware features).

The target operating system(s) included in the various images 22A-22B on the image repository system 12 may include a device driver used to access the local storage on the target system (referred to herein as the "local storage (LS) device driver"). The LS device driver may be configured, in response to a read request for a block from the local storage generated by the target operating system and/or applications, to read the requested block from the local storage in the target system, if the block is present in the local storage. If the block is not present, the LS device driver may fetch the requested block from the image repository system 12. Accordingly, the provisioning software 20, when provisioning a target system 10 with an image 22A, may not copy all of the image blocks comprising the image to the local storage 18A of the target system 10. The blocks not stored in the local storage 18A may be copied from the image 22A as requested. Viewed in another way, execution of the set of software resources may be initiated on the target

system 10 prior to copying at least some of the blocks comprising the image 22A to the target system 10. Accordingly, the time to copy blocks from the image repository system 12 to the target system 10 may not occur during provisioning, and thus provisioning may be completed relatively rapidly. In some cases, one or more blocks in the image 22A-22B may be infrequently accessed, and thus the software resources in the image 22A-22B may execute on the target system 10 for sometime before accessing the infrequently accessed blocks. If a block is not accessed at all, that block may not be transmitted from the image repository system 12 to the target system 10. Furthermore, some blocks may be written before they are read. Such blocks may not be transferred from the image repository system 12 to the target system 10.

While the LS device driver in the target operating system may cause the blocks to be copied in response to attempts to read the blocks, there is a time period in this embodiment, during boot and prior to the initialization of the LS device driver, in which accesses to the local storage may occur. The device driver may be "initialized" if it has been loaded into memory in the target system 10 and is ready to execute to handle accesses to the local storage 18A. For example, before the target operating system is itself initialized, firmware on the target system 10 may be executing to initialize various hardware in the system 10. The firmware may access the local storage 18A prior to the LS device driver being initialized. Generally, firmware may comprise software that is provided with the corresponding computer system, generally stored in non-volatile memory such as read-only memory (ROM), flash memory, battery-backed random access memory (RAM), etc. For example, in PC system, the basic input/output system (BIOS) code may comprise firmware.

When provisioning the target system 10 with a set of software resources, a virtual machine may be established on the target system 10. The virtual machine may have the same configuration as the target system 10, and may be monitored by a virtual machine monitor provided by the provisioner software 20. The virtual machine monitor may trap

accesses by the virtual machine to the local storage 18A. The LS device driver, or code similar to the LS device driver, may copy blocks that are being read and which are not in the local storage 18A to the local storage 18A. Once the target operating system (within the virtual machine) initializes the LS device driver, the virtual machine monitor may terminate and the target operating system may execute natively on the target system 10 (that is, not within a virtual machine).

In some embodiments, the state of the software resources on the target system 10 may be captured as an image on the image repository system 12. The capturing of the state of a set of software resources in an image is referred to as snapshotting. Snapshotting may be performed to create a new image of the software resources on the image repository system 12. Such an image may be created to provide a "golden" image for installation on any target system 10 that is to be provisioned with the same software resources. Alternatively, an image may be snapshot if the target system 10 is to be reprovisioned with a different set of software resources (a different image from the image repository system 12), and the current image is expected to be returned to the target system 10 later (or is expected to be activated on another target system).

In some embodiments, the LS device driver may track which blocks within the image have been written by the target system 10. When a snapshot is desired, only the blocks that have been written may be transmitted by the target system 10 to the image repository system 12. The remaining blocks may be copied from the current image on the image repository system 12 (or the current image may be updated with the blocks provided by the target system 10). The target system 10 may be able to determine which blocks are updated without communication from the image repository system 12. In other embodiments, the LS device driver may track which blocks have been updated with respect to multiple image repository systems.

In some embodiments, provisioning of a target system may be performed from

more than one image repository system. For example, the image to be provisioned may be striped across two or more image repository systems for performance reasons. Alternatively, the image may be stored on each of two or more image repository systems to provide high availability of the image (e.g. if one or more of the image repository systems is down when a block is requested by the target system.

As used herein, an "image" of a set of software resources may comprise the blocks that store the set of software resources. That is, the content of the image may be the same as the content of the set of software resources on the target system after provisioning. In some embodiments, the blocks in the image may be arranged, relative to each other, in the same fashion as the blocks are arranged in the local storage 18A when installed thereon. In other embodiments, any arrangement of the blocks in the image may be used. For example, the blocks may be compressed, stored in consecutive blocks in the repository storage, etc. The image repository may map block identifiers provided by the LS device driver to corresponding blocks in the image. In one exemplary embodiment, the VENTI approach may be used in which each block is stored in a location defined by that block's unique checksum. A "block" may be the unit of storage for the local storage device. That is, the block may be the minimum amount of storage that may be read/written. For example, a block may be 4 kilobytes in size, in some embodiments, although larger or smaller blocks may be defined in other embodiments.

As used herein, a virtual machine comprises any combination of software, one or more data structures in memory, and/or one or more files stored in a local or shared storage. The virtual machine represents the software and hardware used during execution of a given application or applications. Thus, a virtual machine may include a virtual CPU, virtual I/O devices, etc. The virtual machine in which an application is executing may encompass the entire system state associated with an application. A virtual machine monitor may be implemented which monitors execution of the virtual machine, may trap various events occurring within the virtual machine, and may emulate the events. For

example, in the present embodiment, the virtual machine monitor may trap access to the local storage and may use the LS device driver, or software similar to the LS device driver, to perform the access in the local storage or request the accessed block or blocks from the image repository system 12.

The provisioner software 20 may comprise instructions which, when executed, provision target systems with images from the image repository. In some embodiments, the provisioner software 20 may be similar to the OpForce™ product available from VERITAS Software Corporation.

The image repository system 12 may store various images, each image corresponding to a set of software resources that the provisioner software 20 may provision onto a target system. Generally, as used herein, an image repository system may comprise a computer system having one or more images stored in its local storage for possible provisioning onto various target systems. For example, in some embodiments, the image repository system may make the local storage 18B (or the portion thereof storing the images 22A-22B) remotely mountable according to NFS or CIFS. Any other network file system may be used in other embodiments.

The execution hardware 16A-16B may generally comprise hardware used to execute various software on the systems 10 and 12, respectively (e.g. the provisioner software 20 on the system 12). For example, the execution hardware may include one or more processors designed to execute the instructions that comprise the software. The execution hardware may further include local memory in the node (e.g. random access memory (RAM)) and circuitry for interfacing to the network 14.

The network 14 may comprise any network technology in various embodiments. The network 14 may be a local area network, metropolitan area network, wide area network, intranet network, Internet network, wireless network, or any other type of

network or combinations of the above networks. The network 14 may be designed to be continuously available (although network outages may occur), or may be intermittent (e.g. a modem connection made between a computer system in a user's home and a computer system in a user's workplace). Any network media may be used. For example, the network 14 may comprise an Ethernet network. Alternatively, the network may comprise a token ring network, a SAN, etc.

As used herein, the term "local storage" of a computer system may comprise any storage that is privately accessible by the computer system. The storage may be included in the computer system, or coupled to the computer system and external to the computer system. If another computer system is to access data in the local storage, the other computer system must communicate with the computer system, which may access the data on behalf of the other computer system. Local storage may include any type of non-volatile storage, such as fixed or removable disks using magnetic or optical media (e.g. hard drives, floppy disks, compact disc-read only memory (CD-ROM), digital versatile disk ROM (DVD-ROM), CD-recordable (CD-R), CD-Rewritable (CD-RW), DVD-R, DVD-RW, non-volatile random access memory (RAM) such as flash memory, battery-backed RAM, solid state storage, etc.).

While local storage is used as an example of the storage to which the set of software resources for a target system are provisioned, other embodiments may use non-local storage. Generally, any storage from which the target system is configured to boot may be provisioned with the set of software resources, using the mechanisms described herein. For example, the storage may be local storage, SAN storage, iSCSI storage, or any combination of the above storage. While local storage will continue to be used as an example in this disclosure, it is contemplated that any storage from which the target system is configured to boot may be used in other embodiments (along with, e.g., a corresponding device driver similar to the LS device driver described herein, virtual machine technology, in some embodiments, etc.).

Turning now to Fig. 2, a flowchart is shown illustrating one embodiment of a method for provisioning a target system (e.g. the target system 10 in Fig. 1). One or more blocks shown in Fig. 2 may be implemented in various software, as described. That is, the software may comprise instructions which, when executed, implement the functions described for the corresponding blocks.

The target system 10 may boot, and booting may include sending a remote boot request on the network 14 (reference numeral 30). For example, the firmware on the target system 10 may be configured to transmit the remote boot request. In some embodiments, the target system 10 may be a PC system and the remote boot request may comprise a PXE boot request. A PXE boot request may comprise a specially formatted packet transmitted on the network 14 that identifies the booting computer system (e.g. by the media access controller (MAC) address in the network interface controller in the booting computer system) and indicates that a boot is occurring. The booting computer system may timeout after a specified interval and booting may continue with other bootable devices (e.g. fixed or removable disk drives in the booting computer system or coupled thereto). In other embodiments, other types of remote boot protocols may be used. For example, computer systems available from Sun Microsystems, Inc. (Santa Clara, CA) may support a net boot protocol. As used herein, a "remote boot request" may comprise any communication from a booting computer system, transmitted external to the computer system in order to receive boot code from a remote system or device. A response to the remote boot request may include the boot code, or may indicate where the boot code may be located (within the booting computer system or external to the booting computer system).

The provisioner software 20 on the image repository system 12 may detect the remote boot request, and may determine that the target system 10 is to be provisioned with one of the images 22A-22B. Optionally, the provisioner software 20 may customize

the image in the image repository (reference numeral 32). Customizations may include inserting the hostname of the target system 10, the IP address of the target system 10, the system ID of the target system 10, the cluster ID of the target system 10, any other host-specific values, etc. into the image. The provisioner software 20 responds to the remote boot request by transmitting a provisioner operating system (O/S) to the target system 10 (reference numeral 34). For example, the OpForce™ product includes an Active O/S as the provisioner O/S. The Active O/S may be a memory-resident operating system derived from the Linux code base. However, other embodiments may use other operating systems. In some embodiments, the provisioner O/S may be completely memory-resident on the target system 10. That is, the provisioner O/S may not require storage on the local storage 18A, and may execute out of the computer system's memory. The provisioner O/S may include the virtual machine monitor mentioned above, as well as a virtual machine with virtual hardware that matches the target system 10. Additionally, the provisioner O/S may include the LS device driver mentioned above, or code similar in operation thereto.

The target system 10 receives the provisioner O/S and boots with the provisioner O/S (reference numeral 36). The provisioner O/S starts the virtual machine on the target system 10 (reference numeral 38). The virtual machine monitor (VMM) included in the provisioner O/S begins monitoring for accesses to the local storage 18A. Other events may be monitored as well, if desired, in various embodiments. If the VMM detects a local storage access (decision block 40 - "yes" leg), the VMM may trap to the LS device driver in the provisioner O/S to perform the access (reference numeral 42). If the virtual machine has not yet initialized the LS device driver within the operating system that is included in the image (decision block 44 - "no" leg), the VMM continues monitoring for local storage 18A accesses. Once the virtual machine has initialized the LS device driver (decision block 44 - "yes" leg), the VMM may exit and the target operating system may be active as the native operating system on the target computer system 10 (reference numeral 46). Even if the target system 10 is subsequently rebooted, the virtual machine

may not be required to be established again since the blocks of the local storage 18A that are accessed prior to initializing the LS device driver have been copied to the local storage 18A and may be accessed locally. In other embodiments, firmware on the target system 10 may be configured to perform the functions assigned to the provisioner O/S.

Turning now to Fig. 3, a block diagram is shown illustrating one embodiment of the target system 10 and the image repository system 12 at a point in time in the operation of the flowchart of Fig. 2 after the virtual machine has been established on the target system 10 but before the LS device driver has been initialized in the target operating system. The target system 10 has the provisioner O/S 50 described above, including the VMM 52 and the LS device driver 54, executing on the execution hardware 16A (and memory resident, in one embodiment). Additionally, the virtual machine 56 is active, executing the target operating system 58, which includes the LS device driver 54. The target O/S 58 and the LS device driver 54A within the target O/S 58 are shown in dotted form, since the blocks storing this code may not yet have been fetched from the image repository system 12 and stored in the local storage 18A.

Shown in the local storage 18A are several blocks from the image 22A (which, in this example, may be the image corresponding to the target O/S 58 and any other desired software resources, not shown). For example, IB0, IBN, and IB7 may be stored in the local storage 18A. As mentioned above, the LS device driver 54 may track which blocks are stored in the local storage 18A. For example, in the illustrated embodiment, the LS device driver 54 may maintain an LS bitmap 60 which identifies which blocks are stored in the local storage 18A. When the LS device driver 54 fetches one or more blocks from the image repository system 12 and stores the blocks in the local storage 18A, the LS device driver 54 may update the LS bitmap 60 to indicate that the fetched blocks are in the local storage 18A.

In one embodiment, the LS bitmap 60 may include an entry for each block in the

local storage 18A. The entry may include a bit (referred to as the read bit with regard to Fig. 10) indicating whether or not the block is present in the local storage 18A. For example, the bit may be set to indicate that the block is present and clear to indicate that the block is not present. Alternatively, the opposite meanings of the set and clear states of the bit may be used, or other encodings may be used. Additionally, each entry may include one or more bits for indicating whether or not the corresponding block has been written (e.g. the W0 to WN bits in Fig. 10), as described in more detail below.

Fig. 4 is a block diagram illustrating one embodiment of the target system 10 and the image repository system 12 at a point in time in the operation of the flowchart of Fig. 2 after the virtual machine has exited. The target O/S 58 is shown, no longer executing with a virtual machine but instead executing directly on the execution hardware 16A as a "real" machine. The target O/S 58 is shown in dotted form to indicate that portions of the target O/S 58 may not yet be resident on the local storage 18A. The LS device driver 54 may, however, be fully resident on the local storage 18A at this point. The target O/S 58 may optionally include a background replicator 70 and/or a background provisioner 72. Executing on the target O/S 58 may be a provisioner agent 74. Additional blocks from the image 22A may be resident on the local storage 18A, having been stored therein by the LS device driver 54.

The background provisioner 72 may be a background process (a process that executes when other portions of the target O/S 58 and applications executing thereon are idle) that may fetch blocks from the image repository system 12 and store them in the local storage 18A, without any corresponding request from the target O/S 58 or applications. The background provisioner 72 may also update the LS bitmap 60 to indicate that the blocks are present in the local storage 18A. The background provisioner 72 may be used to complete the fetching of the blocks in the image 22A, so that the target system 10 may not be dependent on the availability of the network 14 for fetching blocks at a later time. In other embodiments, the image repository system 12 may push blocks to

the target system 10, and the background provisioner 72 may receive the blocks and store them in the local storage 18A. Additional details of one embodiment of the background provisioner 72 are provided below with regard to Fig. 6.

The background replicator 70 may be a background process that may replicate modified blocks to the image repository system 12. In this manner, when a snapshot is desired, some of the modified blocks may already have been provided by the target system 10 to the image repository system 12. Additional details of one embodiment of the background replicator 70 are provided below with regard to Fig. 9.

The provisioner agent 74 may comprise software that is designed to communicate with the provisioner software 20. Particularly, the provisioner agent 74 may be used in one embodiment of snapshotting the target system 10, described in more detail below.

It is noted that, while a bitmap 60 is shown in Figs. 3 and 4 as the mechanism for recording which blocks are present in the local storage 18A and/or which blocks have been written by the target system 10, other embodiments may record such blocks in any fashion. For example, one or more lists may be maintained of which blocks are present and which have been written to. In other embodiments, each block may include an indication of whether or not it is present and whether or not it has been written to. As used herein, a "map" may comprise any data structure capable of tracking which blocks are present and which have been written. Generally, the LS device driver 54 may employ any mechanism for recording or tracking which image blocks are present in the local storage and/or modified with respect to the image on the image repository system.

Fig. 5 is a flowchart illustrating operation of one embodiment of the LS device driver 54 in response to a read request generated by the target O/S 58, or applications executing thereon. That is, the LS device driver 54 may include instructions which, when executed, implement the function of the flowchart shown in Fig. 5.

The LS device driver 54 may check the LS bitmap 60 for the requested block (reference numeral 80). If the LS bitmap 60 indicates that the block is present in the local storage 18A (decision block 82 - "yes" leg), the LS device driver 54 may read the block from local storage (reference numeral 84) and return the block to the requestor (reference numeral 86). On the other hand, If the LS bitmap 60 indicates that the block is not in the local storage 18A (decision block 82 - "no" leg), the LS device driver 54 may fetch the block from the image repository system 12 (reference numeral 88), store the block in the local storage 18A (reference numeral 90), update the LS bitmap 60 to indicate that the block is stored in the local storage 18A (reference numeral 92), and return the block to the requestor (reference numeral 86). As mentioned above, the block may be fetched from one of one or more repositories, each of which may store the image or a portion of the image, in various embodiments.

In one embodiment, the image repository system 12 may make the various images 22A-22B mountable under NFS, CIFS, or another network filesystem. The target system 10 may mount the image 22A, and fetching the blocks from the image repository system 12 may occur through the network filesystem mechanism. Other embodiments may use any communication mechanism to transfer blocks between the image repository system 12 and the target system 10. For example, transfer over a network, transfer between devices in a SAN, transfers between iSCSI devices, or any combination of the above may be used.

While the description above refers to a read request for a block, a read request may be for multiple blocks, in some embodiments. In such embodiments, the device driver 54 may fetch any blocks that are not in the local storage 18A from the image repository system 12 and store them in the local storage 18A, and may read the requested blocks from the local storage 18A for return to the requestor.

Fig. 6 is a flowchart illustrating operation of one embodiment of the background provisioner 72. That is, the background provisioner 72 may include instructions which, when executed, implement the function of the flowchart shown in Fig. 6.

Optionally, the background provisioner 72 request one or more blocks from the image repository system 12 (reference numeral 100). The background provisioner 72 may consult the LS bitmap 60 to determine which blocks to request (i.e. blocks that are not already stored in the local storage 18A). In other embodiments, the image repository system 12 may transmit the blocks to be stored, rather than the background provisioner 72 requesting them. Generally, the background provisioner 72 may request or receive blocks from the image repository system 12 independent of requests received by the device driver 54.

If the background provisioner 72 receives a block or blocks from the image repository system 12 (decision block 102 - "yes" leg), the background provisioner 72 may store the received blocks in the local storage 18A (reference numeral 104). In some embodiments, the background provisioner 72 may check the LS bitmap 60 to ensure that the LS device driver 54 has not written the block (in response to a write request from the target O/S 58 or applications executing thereon), to avoid overwriting the block in the local storage 18A. The background provisioner 72 may also update the bitmap to indicate that the received blocks are now present in the local storage 18A (reference numeral 106).

The background provisioner 72 may determine if all the blocks from the image 22A have been installed in the local storage 18A (decision block 108). If not (decision block 108 - "no" leg), the background provisioner 72 may continue requesting and receiving blocks from the image repository system 12. If all blocks from the image repository system 12 have been installed (decision block 108 - "yes" leg), the background provisioner 72 may exit. In some embodiments, the background provisioner 72 may communicate to the LS device driver 54 that all blocks have been stored in the local

storage 18A. The LS device driver 54 may record an indication that all blocks are present, and may cease checking the LS bitmap 60 for read requests.

Turning now to Fig. 7, a flowchart is shown illustrating one embodiment of a method for snapshotting a provisioned target system (e.g. the target system 10 in Fig. 1). One or more blocks shown in Fig. 7 may be implemented in various software, as described. That is, the software may comprise instructions which, when executed, implement the functions described for the corresponding blocks.

Optionally, the provisioner software 20 may request that the provisioner agent 74 cause a reboot of the target system 10 (reference numeral 110). Such a request may be made, for example, if a fully consistent snapshot is desired. Some software may update blocks on the local storage 18A, but may not commit the updates until a later time (e.g. journaled filesystems, transaction-based systems such as databases, etc.). In effect, the order of updates to the blocks may be important to the consistency of the image. However, order may not be determined from the LS bitmap 60. Block 110 may be eliminated if a fully consistent snapshot is not desired, or if the software executing on the target system 10 does not execute in such a way that inconsistencies are possible.

The provisioner software 20 may request the modified blocks from the provisioner agent 74 (or, alternatively, directly from the target O/S 58) (reference numeral 112). Using the modified indications in the LS bitmap 60 for each block, the provisioner agent 74 or the target O/S 58 may determine which blocks have been modified, and may transmit the blocks to the provisioner software 20 (reference numeral 114). The provisioner software 20, after successfully receiving the modified blocks, may transmit an acknowledge to the provisioner agent 74 or the target O/S 58 (reference numeral 116). In response to the acknowledge, the provisioner agent 74 may update the bitmap to indicate that the blocks are not modified (reference numeral 118).

In some embodiments, the LS bitmap 60 may include multiple indications for each block to indicate that the block has been updated. Such embodiments may be used if replication/snapshotting to multiple computer systems is desired. When a given computer system requests modified blocks, the corresponding indications for each block may be used to determine if the block is modified and the corresponding indications may be updated to indicate not modified.

It is noted that the same image (e.g. image 22A) may be used to provision multiple separate systems concurrently, provided that snapshotting of each system is maintained in a fashion that the snapshots may be uniquely determined (e.g. using a copy on write approach).

Fig. 8 is a flowchart illustrating operation of one embodiment of the LS device driver 54 in response to a write request generated by the target O/S 58, or applications executing thereon. That is, the LS device driver 54 may include instructions which, when executed, implement the function of the flowchart shown in Fig. 8.

The LS device driver 54 may store the block supplied by the requestor in the local storage 18A (reference numeral 120). The LS device driver 54 may update the LS bitmap 60 to indicate that the block is present (in case it was not present previously) and modified (reference numeral 122). Optionally the LS device driver 54 may transmit the block to the image repository 12 for storage (reference numeral 124). If the block is transmitted, the LS device driver 54 may update the LS bitmap 60 to indicate not modified responsive to receiving an acknowledge from the image repository system 12. The LS device driver 54 may transmit the block synchronously or asynchronously. In yet another option, automatic replication software (such as Veritas Volume Replicator or Veritas Storage Replicator products available from VERITAS Software Corporation) may be used to replicate the modified blocks to the image repository system 12. In yet other alternatives, the background replicator 70 may be used.

While the description above refers to a write request for a block, a write request may be for multiple blocks, in some embodiments. In such embodiments, the device driver 54 may store each block in the local storage 18A and update the LS bitmap 60 accordingly (and optionally transmit the block to the image repository system 12).

Turning now to Fig. 9, a flowchart is shown illustrating operation of one embodiment of the background replicator 70. That is, the background replicator 70 may include instructions which, when executed, implement the function of the flowchart shown in Fig. 9.

The background replicator 70 may check the LS bitmap 60 to determine if there are any modified blocks in the local storage 18A (reference numeral 130). If there are no modified blocks (decision block 132 - "no" leg), the background replicator 70 may be idle. If there are modified blocks (decision block 132 - "yes" leg), the background replicator 70 may transmit one or more modified blocks to the image repository system 12 (reference numeral 134). The background replicator 70 may receive the acknowledgement from the image repository system 12 (reference numeral 136), and may update the LS bitmap 60 to indicate that the transmitted blocks are not modified responsive to the acknowledgement (reference numeral 138). As mentioned above, replication may be performed to more than one repository, in some embodiments.

As mentioned above, if the target system 10 is not rebooted prior to copying modified blocks, it may be difficult to generate a consistent snapshot. In some embodiments, the volume manager or filesystem executing on the target system 10 may be modified to insert calls to the LS device driver 54 when a transaction is being completed (e.g. after the filesystem metadata updates, such as directory updates, have been completed). The calls from the volume manager or filesystem may be used to mark when a consistent state may be available, which may be used to determine when a

consistent snapshot may be created.

It is noted that, in some embodiments, the provisioner software 20 may be configured to speculatively include one or more additional blocks in response to a fetch of blocks by the LS device driver 54. In this manner, copying of blocks to the target system 10 may be accelerated. In some embodiments, the provisioner software 20 may be configured to track the pattern of requests from the LS device driver 54 to predict the next most likely blocks to be fetched, and the predicted blocks may be included in response to the fetch (along with the requested blocks).

Turning now to Fig. 10, a block diagram illustrating one embodiment of the LS bitmap 60 is shown. The LS bitmap 60 may include an entry for each block in the local storage 18A (e.g. entries 140A-140N in Fig. 10). Entry 140C is shown in exploded view, and includes a read bit (R) and may optionally include one or more write bits (W0, W1, etc. through WN). The read bit may indicate whether or not the block is present in the local storage 18A (as described previously). One of the write bits may indicate whether or not the block is modified with respect to the corresponding block in the image on the image repository system 12. For example, if the write bit is set, the block may be modified with respect to the corresponding block in the image and if the write bit is clear, the block may not be modified with respect to the corresponding block in the image. Alternative embodiments may use the opposite definition of the set and clear states, or another encoding. Multiple write bits may be included, in some embodiments, to track whether a given block is modified with respect to multiple image repository systems (or other computer systems that may be used to make replicated copies of the images).

It is noted that, while the above embodiment includes an LS device driver that tracks both which blocks are present (to permit provisioning without copying all the blocks of the image) and which blocks have been written (to permit snapshotting as described herein), other embodiments may implement only tracking which blocks are

present (and the corresponding provisioning features) or only which blocks have been written (and the corresponding snapshotting features), as desired.

Turning now to Fig. 11, a block diagram of a computer accessible medium 150 is shown. Generally speaking, a computer accessible medium may include any media accessible by a computer during use to provide instructions and/or data to the computer. For example, a computer accessible medium may include storage media such as magnetic or optical media, e.g., disk (fixed or removable), CD-ROM, or DVD-ROM, CD-R, CD-RW, DVD-R, DVD-RW, volatile or non-volatile memory media such as RAM (e.g. synchronous dynamic RAM (SDRAM), Rambus DRAM (RDRAM), static RAM (SRAM), etc.), ROM, etc., as well as media accessible via transmission media or signals such as electrical, electromagnetic, or digital signals, conveyed via a communication medium such as a network and/or a wireless link. The computer accessible medium 150 in Fig. 11 may be encoded with one or more of the provisioner agent 74, the provisioner software 20, the VMM 52, the LS device driver 54, the provisioner O/S 50, the images 22A-22B, the target O/S 58, the background provisioner 72, and/or the background replicator 70. Each of the provisioner agent 74, the provisioner software 20, the VMM 52, the LS device driver 54, the provisioner O/S 50, the target O/S 58, the background provisioner 72, and the background replicator 70 may comprise instructions which, when executed, implement the functionality described herein for the software. Generally, the computer accessible medium 150 may store any set of instructions which, when executed, implement a portion or all of the flowcharts shown in one or more of Figs. 2 and 5-9.

It is noted that, while the provisioner agent 74, the provisioner software 20, the VMM 52, the LS device driver 54, the provisioner O/S 50, the target O/S 58, the background provisioner 72, and the background replicator 70 have been described as software executing on various systems, one or more of the above may be implemented partially in software and partially in hardware in the respective systems, or wholly in hardware in the respective systems, in various embodiments.

Numerous variations and modifications will become apparent to those skilled in the art once the above disclosure is fully appreciated. It is intended that the following claims be interpreted to embrace all such variations and modifications.